Dependent method claim **12** requires that the method of the second real multiplication of claim **9** not use the product of the first real number and the second real number that is calculated by the first real multiplication method. This means that the method of the second real multiplication must use at least one member of the first set of intermediate terms. Claim **12** is intended to cover methods for parallel computation of the two products.

Dependent method claim **13** restricts method claim **9** by further including addition of the first product and the second product to a first product sum which is not a desired product of two numbers.

Dependent method claim **14** restricts method claim **9** by further including addition of the first product to a first product sum and second addition of the second product to a second product sum. The two product sums are separate. The method of this embodiment of the invention covers shared multiplication in signal processing transforms such as computing the contribution of one transform input to two separate transform outputs.

Independent method claim **15** comprises multiplication to produce a first product and a second product. The first product is the product of a first number and a second number. The third product is the product of the first number and a third number. The first number is represented in a first finite-precision numeric format. The second number is represented in a second finite-precision numeric format. The third number is represented in a third finite-precision numeric format. The multiplication method uses at least one of the calculation results used in computing the first product for computing the second product as well.

Dependent method claim **16** restricts method claim **15** by requiring that the second product not equal the product of the first number and the complex conjugate of the second number except when the first number is zero or the second number is equal to

the complex conjugate of the third number. Also, the second product is not equal to the product of the second number and the complex conjugate of the first number except when the first number is zero or the first number is real and the second number is equal to the third number.

The restrictions imposed in claim **16** on the products of claim **15** are intended to emphasize that a multiple-output multiplication method with shared computation is not limited to computing a first product which is the product of two complex numbers and a second product which is the product of one of the complex numbers and the complex conjugate of the other complex number. Other complex number values may allow sharing of computation. Also, particular representations of other complex number values in particular finite-precision numeric formats may allow sharing of computation.

DESCRIPTION - FIG 2

Thus far the present invention has been discussed in terms of multipliers or multiplication of numbers in finite-precision numeric formats. Below is a specific example of two number values and their representations in a finite-precision numeric format. The discussion clarifies how a representation of one number can have common properties with a representation of another number.

In digital signal processing, every number is stored in a finite-precision numeric format. The format is defined by a finite number of representation elements and a mapping between numbers and representation element values. While it is possible to have multiple types of representation elements, it is common to use one type of representation element, in particular binary representation elements, or bits, which can take on two possible values. Common binary mappings include signed integer, unsigned integer, floating point, and twos complement, among others.

Fig 2 shows the 16-bit twos complement representations of $\sin(2\pi / 32)$ and $\sin(2\pi / 64)$, two numbers which are possible weights used in computing a discrete Fourier

transform. There are six non-zero bits in the 16-bit twos complement representation of sin(2 π / 64), including a fifth bit of sin(2 π / 64) **34**, a sixth bit of sin(2 π / 64) **36**, a fifteenth bit of sin(2 π / 64) **50**, and a sixteenth bit of sin(2 π / 64) **52**. The 16-bit decimal value of sin(2 π / 64) **62** is 0.097991 to six decimal places. The desired decimal value of sin(2 π / 64) **46** is 0.098017.

There are seven non-zero bits in the 16-bit twos complement representation of sin(4 π / 64). Among these are a fourth bit of sin(4 π / 64) **38**, a fifth bit of sin(4 π / 64) **40**, a ninth bit of sin(4 π / 64) **54**, a tenth bit of sin(4 π / 64) **56**, an eleventh bit of sin(4 π / 64) **58**, and a twelfth bit of sin(4 π / 64) **60**. The 16-bit decimal value of sin(4 π / 64) **64** is 0.195068 to six decimal places. The desired decimal value of sin(4 π / 64) **48** is 0.195090 to six decimal places. With more bits, or higher precision, the actual value more closely matches the desired value.

Consider multiplying the 16-bit twos complement representation sin(2 π / 64) by an arbitrary number. One might first compute a first partial product consisting of the contribution to the product of fifteenth bit of sin(2 π / 64) **50** and sixteenth bit of sin(2 π / 64) **52** multiplied by the arbitrary number. These two bits are adjacent bits that each have value 1. Fifth bit of sin(2 π / 64) **34** and sixth bit of sin(2 π / 64) **36** are also two adjacent bits that each have a value 1. As an intermediate term, the first partial product could be shifted ten binary places to obtain a second partial product consisting of the contribution of fifth bit of sin(2 π / 64) **34** and sixth bit of sin(2 π / 64) **36** to the product of this representation of sin(2 π / 64) and the arbitrary number. This is the idea behind shared computation in the constant multiplier techniques of US Patent 4,868,778 and US Patent 5,841,684.

Next, consider multiplying the 16-bit twos complement representation of sin(4 π / 64) by the arbitrary number. In the 16-bit twos complement representation of sin(4 π / 64), fourth bit of sin(4 π / 64) **38** and fifth bit of sin(4 π / 64) **40** are two adjacent bits that each have a value 1. Also, ninth bit of sin(4 π / 64) **54** and tenth bit of sin(4 π / 64) **56,**